

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Application of:  
John Wainright

Serial No.: 09/426,143

Filed: October 22, 1999

For: SPECIFYING OPERATIONS  
TO BE APPLIED TO THE  
ATTRIBUTES OF A SET OF  
OBJECTS

§ Confirmation No.: 1474  
§  
§ Group Art Unit: 2628  
§  
§ Examiner: Chante E. Harrison  
§  
§  
§  
§  
§

MAIL STOP APPEAL BRIEF-PATENTS  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**CERTIFICATE OF MAILING OR TRANSMISSION**

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Chante E. Harrison, or electronically transmitted via EFS-Web, on the date shown below:

March 24, 2008  
Date

/Jon K. Stewart/  
Jon K. Stewart

**REPLY BRIEF**

Dear Sir:

Applicant submits this Reply Brief to the Board of Patent Appeals and Interferences in response to Examiner's Answer mailed on January 23, 2008. While Applicants' maintain each of the arguments submitted in Applicants' previously submitted Appeal Brief, Applicants make the following further arguments in light of the Examiner's Answer.

## ARGUMENTS

The Examiner continues to suggest that *Merrill*, ¶¶ 168-169 and ¶ 326 disclose a “method of executing an operation on a set of graphical components” that includes the claimed step of “detecting that a statement contains an operation identifier, pattern-matching criteria, and attribute identifier.” And that *Merrill*, ¶ 340 discloses the claimed step of “executing said statement by identifying said set of graphical components associated with identifiers that satisfy said pattern matching criteria....” Respectfully, Applicants disagree.

*Merrill*, ¶¶ 168-169, describe aspects of a particular programmatic object (specifically, an “agent” object) included in as part of “animation server.” As described, the animation server is “implemented as an OLE<sup>1</sup> Server.” *Merrill*, ¶ 147. As is well known, an “object” encapsulates a set of data values and a set of operations on those data values. Programs create instances of a given object and invoke the operations supported by the object to accomplish a computing task. Collectively, the set of operations define an “interface” to the object. *See, e.g., Merrill*, ¶¶ 123-130 (providing a general description of object oriented programming techniques).

As disclosed in *Merrill*, one of the programmatic objects supported by the “animation server” is an “Agent” object. *Merrill* describes this object as follows:

Clients of the animation server access its animation services using the methods, properties and events of the agent object’s interface. The methods of the agent object include a number of functions to control the playback of an animation. Example methods include: Play, GestureAt, MoveTo, Stop and Speak.

*Merrill*, ¶ 156. Also, *Merrill* discloses that clients invoke the “Speak” method “to instruct the server to generate speech output for a specified text string.” *Merrill*, ¶ 162.

Further, *Merrill* discloses that a tag may be embedded within the “specified text string” of a call to the “Speak” method. The tag is used to instruct the animation server to notify the client when the bookmark is reached when generating speech output. *Merrill* provides the following example of how a programmer may use the “Speak” method:

Agent1.Speak “Generate a bookmark now\mrk=100.”

*Merrill*, ¶ 169. The call to the “Speak” method is formatted according to well known conventions for accessing a method provided by object’s interface. Specifically, the format of

---

<sup>1</sup> OLE is short for Object Linking and Embedding and refers to a software architecture developed by Microsoft. OLE allows objects from one application to be embedded within another.

“Agent1.Speak” indicates that the animation server should invoke the “Speak” method for an instance of the “agent” object executing on the animation server named “Agent1.” Further, the text “Generate a bookmark now\mrk=100” is passed as a parameter to the “Agent1” instance of the “Agent” object and provides the text used to generate speech output. Thus, in this example, when the “Speak” method is invoked, the animation server generates audio data reciting the words “Generate a bookmark now” and transmits this data audio data to the client for playback. Further, while doing so, the animation server eventually encounters the “mrk=100” following the “\” escape character. As described in *Merrill*, when this occurs, the animation server sends a notification message to the client with a bookmark ID of “100” along with the audio data. Doing so allows the client which invoked the “Speak” method (of the “Agent1” object on the server) to synchronize other events occurring on the client with the audio data returned by the “Speak” method.

The Examiner relies on the described “Speak” method of the “Agent” object of the animation server along with examples of object oriented programming syntax to argue the *Merrill* discloses a “method of executing an operation on a set of graphical components” that includes the claimed step of “detecting that a statement contains an operation identifier, pattern-matching criteria, and attribute identifier.” Specifically, the Examiner suggests:

The Speak Statement is an exemplary script format, such as those illustrated at pp. 19, Para 324-327. The illustrated script statements are used to execute operations on the attributes of objects, which the Applicant refers to as graphical components. For example, pp. 19, Para 326, is a script that would execute an operation, such as adjusting a "value", of an attribute, such as "Property" on an object or graphical component that has the text string pattern "agent".

*Examiner’s Answer*, p. 8. In this case, the Examiner fundamentally misapplies *Merrill* to argue that the “agent” object discloses the claimed “pattern matching criteria.” The “agent” object provides a programmatic object with a defined interface (including the “Speak” method), which may be accessed according to the syntax discussed above, and does not disclose the “pattern matching criteria” included in the claimed detecting step. Further, the Examiner cites the following additional examples of programming syntax for accessing objects:

[0324] To use a method or property in VBScript (or Visual Basic®), the programmer uses the conventional syntax for methods, properties, and events. Examples of this syntax are set forth below:

[0325] agent.object.Method argument

[0326] agent.object.Property=value

[0327] agent.object\_Event (argument as datatype)

*Merrill*, ¶¶ 324-327. Candidly, these passages do not disclose anything other than well known syntax supported by many object oriented programming languages. In fact, *Merrill* itself describes these examples as nothing more than “the conventional syntax for methods, properties, and events.” The “agent.object” is clearly not “pattern matching criteria;” instead, in the context of *Merrill*, it represents the name of an instance of an object running on the “animation server.” To invoke a method of an instance of this object, a programmer would use the format “agent.object.Method.” Similarly, a programmer would use the format of “agent.object.Property=value” to assign value to a data member encapsulated by this object. Ultimately, however, these examples are nothing more than “the conventional syntax for methods, properties, and events.”

Importantly, Applicants do not claim “conventional syntax for methods, properties, and events;” instead, Applicants claim a specific and detailed method for “executing an operation on a set of graphical components” that includes “detecting that a statement contains an operation identifier that specifies said operation, pattern matching criteria, and an attribute identifier that identifies an attribute.” And that further includes “executing said statement by identifying said set of graphical components associated with identifiers that satisfy said pattern matching criteria” and “performing said operation on said attribute of each graphical component in said set of graphical components that satisfy said pattern matching criteria.”

Thus, as the foregoing discussion demonstrates, the Examiner is clearly flawed in arguing:

Merrill teaches a script statement (pp. 19, Para 326) that contains pattern matching criteria, e.g. "agent", an attribute, e.g. "Property", and an operation, e.g. "value"; and executes the statement by identifying the graphical components matching the identifiers (i.e. the text strings) (pp. 13, Para 168-169) and performing the operation (pp. 13, Para 170), such as display of the animated agent object speaking in sync with the displayed text.

*Examiner's Answer*, p .9. Again, the Examiner confuses the name of an instance of an object (namely, “agent”) with the claimed “pattern matching criteria.” Further, the Examiner confuses the claimed “operation identifier that specifies said operation,” with a “value,” which in fact refers to a value assigned to a data member of an object using conventional object oriented syntax. Furthermore, the “text strings” at *Merrill*, ¶¶ 168-169 refer to a parameter passed to the “Speak” method, and specify what should be recited by the animation server (i.e., what text-to-speech should be generated and returned to the client). In no way does this parameter “execute[]

the statement by identifying the graphical components matching the identifiers (i.e. the text strings),” as suggested by the Examiner. That is, the “agent” object is not used to identify a set of “graphical components matching the text strings,” as claimed. Instead, the “agent” object of *Merrill* includes a number of methods, one of them being a “speak” method, which may be passed a parameter specifying what, in fact, should be “spoken.”

Accordingly, for all the foregoing reasons, Applications submit that Claim 1 is patentable over *Merrill*. Independent claims 8, 12, and 18 recite similar limitations and are rejected using the same flawed analysis the Examiner applies to claim 1. Applicants submit, therefore, that claims 8, 12, and 18 are allowable as well.

### **CONCLUSION**

The Examiner errs in finding that claims 1-3, 5, 7-14, 16 and 18-22 are unpatentable over *Merrill* under 35 U.S.C. § 103(a). Withdrawal of the rejection and allowance of all claims is respectfully requested.

Respectfully submitted,

/Jon K. Stewart/

Jon K. Stewart

Registration No. 54,945

Patterson & Sheridan, L.L.P.

3040 Post Oak Blvd. Suite 1500

Houston, TX 77056

Telephone: (713) 623-4844

Facsimile: (713) 623-4846

Attorney for Appellant